Remaining Useful Life Estimation for Predictive Maintenance Applications

Authors: Jyosna Philip, Abhay Sharma, Muthukumar Ganesan, Anchal Sekhri, Khunwana Zeno, Sana Zehra

Objective:

Predictive maintenance, aimed at predicting when equipment will fail and scheduling timely maintenance to prevent unexpected breakdowns, has emerged as a critical field in various industries. The estimation of Remaining Useful Life (RUL) of machinery and equipment is at the forefront of this domain, leveraging advanced data analytics and machine learning techniques to foresee equipment failure and optimize maintenance schedules. Remaining Useful Life (RUL) of a component or a system is defined as the length from the current time to the end of the useful life. Accurate RUL estimation plays a critical role in Prognostics and Health Management (PHM). Data driven approaches for RUL estimation use sensor data and operational data to estimate RUL.

Introduction:

Industrial systems, ranging from small machines to Jet engines, rely on maintenance for their durability. In recent years, companies have started to invest in techniques that can prevent faults in advance. One such technique is Preventive Maintenance [1]. Remaining Useful Life (RUL) of a component or a system is defined as the length from the current time to the end of the useful life [2]. The criteria to define whether the component or system is still usable is already known to the domain experts of the component or system. In industry operational research, there is an interest in modeling methods for RUL estimation given component or system condition and health monitoring information. Prognostic technologies are very crucial in condition based maintenance for diverse application areas, such as manufacturing, aerospace, automotive, heavy industry, power generation, and transportation. While accessing the degradation from expected operating conditions, prognostic technologies estimate the future performance of a subsystem or a component to make RUL estimation. If we can accurately predict when an engine will fail, then we can make informed maintenance decision in advance to avoid disasters, reduce the maintenance cost, as well as streamline operational activities. This works proposes a data driven approach to predict RUL of a complex system when the run-to-failure data is available.

In this work, we propose a comprehensive methodology to analyze the open-source dataset of NASA's Turbofan Jet Engine for predicting Remaining Useful Life (RUL). The project encompasses a series of steps including Exploratory Data Analysis (EDA), Feature Engineering (FE), and the application of multiple machine learning algorithms to develop a robust predictive model. The objective is to provide an in-depth understanding and a high-performing model that can accurately forecast the RUL of jet engines, thereby facilitating predictive maintenance.

Data Preparation:

Data set consists of multiple multivariate time series. The data set is further divided into training and test subsets. Each time series is from a different engine i.e., the data can be

considered to be from a fleet of engines of the same type. Each engine starts with different degrees of initial wear and manufacturing variation which is unknown to the user. This wear and variation is considered normal, i.e., it is not considered a fault condition. The data is contaminated with sensor noise. The engine is operating normally at the start of each time series, and develops a fault at some point during the series. In the training set, the fault grows in magnitude until system failure. In the test set, the time series ends some time prior to system failure. The objective of the competition is to predict the number of remaining operational cycles before failure in the test set, i.e., the number of operational cycles after the last cycle that the engine will continue to operate. Also provided a vector of true Remaining Useful Life (RUL) values for the test data.

CMAPSS Dataset: The data contains 26 columns of numbers, where each row is a snapshot of data taken during a single operational cycle. The columns corresponds to Engine ID, Time in Cycles, Setting1, Setting2, Setting3 and remaining all are sensor data.

Target RUL: The target variable is not explicitly provided in the dataset; therefore, it is inherently calculated by subtracting the number of cycles from the maximum cycle for the corresponding engine. This results in the creation of a new column labelled "Remaining Cycles," which represents the Remaining Useful Life (RUL) of the engine. The health of a system degrades linearly along with time. In practical applications, degradation of a component is negligible at the beginning of use, and increases when component approaches end-of-Life. To better model the Remaining Useful Life changes along with time, in [3] [4], a piece-wise linear RUL target function was proposed, as in Figure below, which limits the maximum RUL to a constant value and then start linear degradation after a certain degree of usage. We set the maximum limit as 130 time cycles for all the engines.

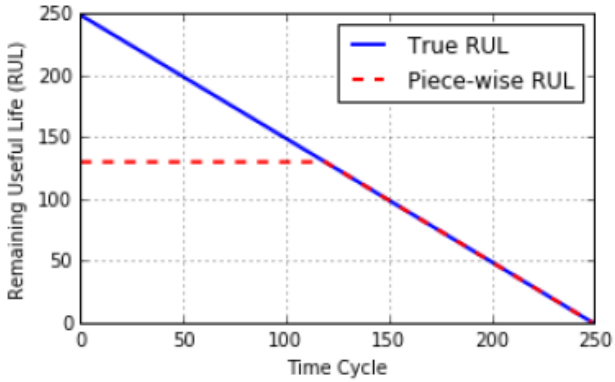
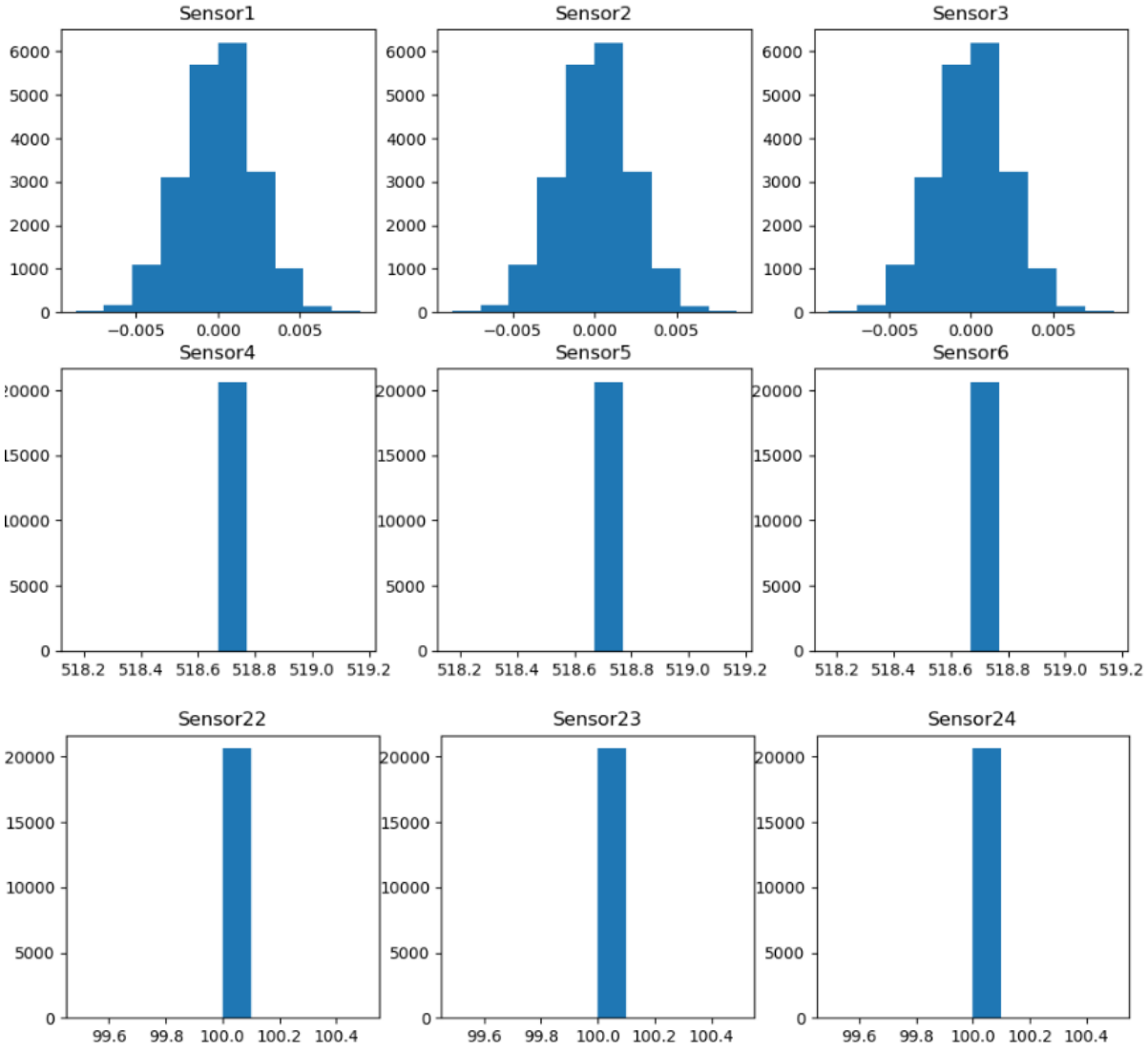


Fig. Piece-wise RUL of the Data Set (Piece-wise maximum RUL is 130 time cycles)

Exploratory Data Analysis:

The data distribution of each variable in the dataset varies significantly. The setting variables and sensor variables include constant, discrete, and continuous data types. As per figure, some sensors, such as sensor4, sensor5, and sensor6, exhibit constant values. In contrast, sensors such as Sensor1, Sensor2, and Sensor3 show a normal distribution. The remaining sensors display skewed distributions, either to the right or left. This diverse range of data distributions necessitates tailored pre-processing techniques to ensure optimal model

performance.



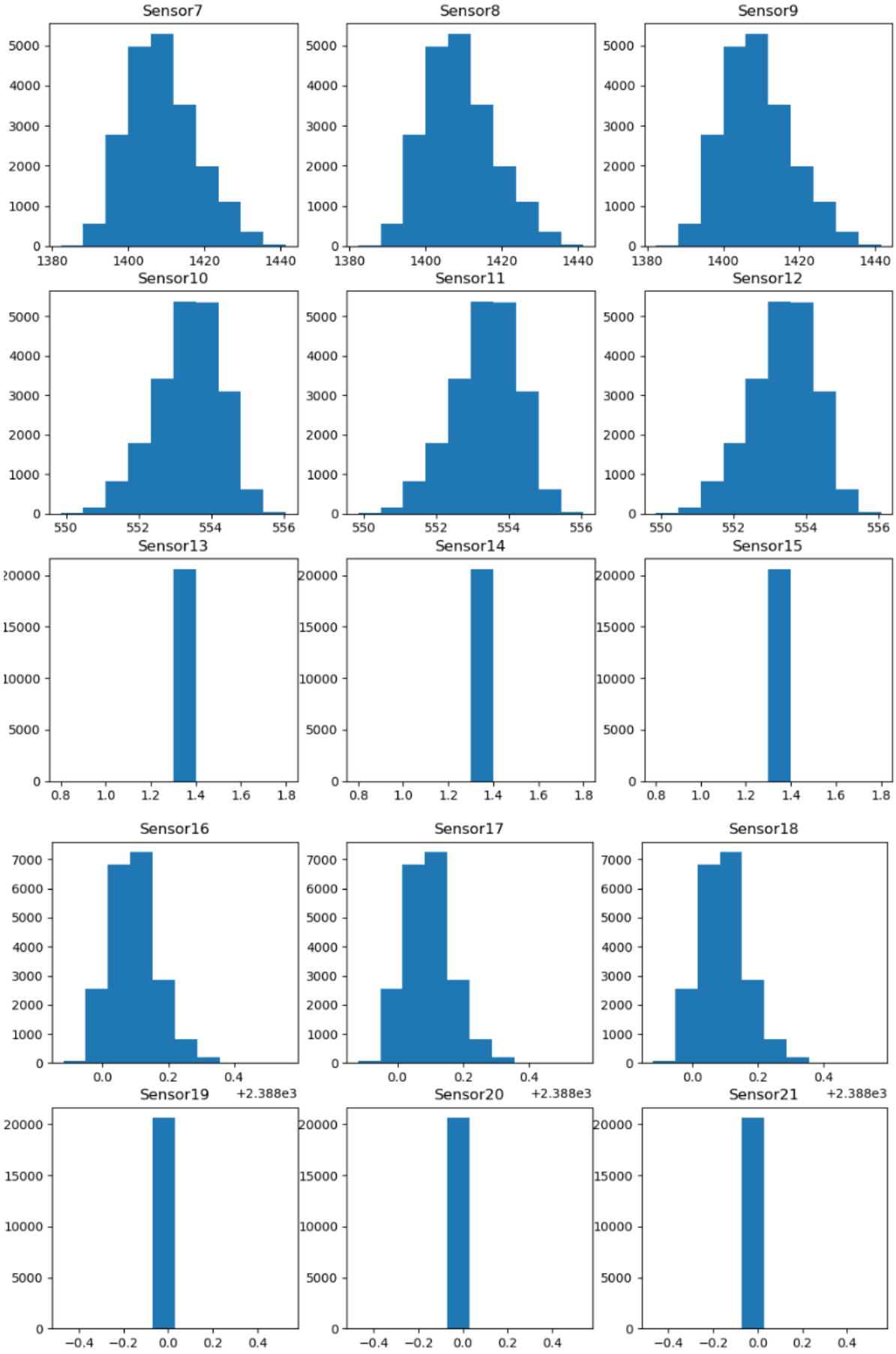


Fig. Frequency Distribution Plot for Sensor Data

Parallel Plots (also known as Parallel Coordinate Plots) [11] are a type of data visualization used to explore and analyse multi-dimensional data. This technique is particularly effective for understanding the relationships between multiple variables and for identifying patterns, clusters, and outliers in high-dimensional datasets. In the NASA Aircraft Engine dataset, a parallel plot was utilized shown in the below figure, where each vertical axis represents a different sensor, and each line corresponds to an observation in the dataset. Dark blue lines represent observations with higher remaining cycles, while yellow lines indicate lower remaining cycles.

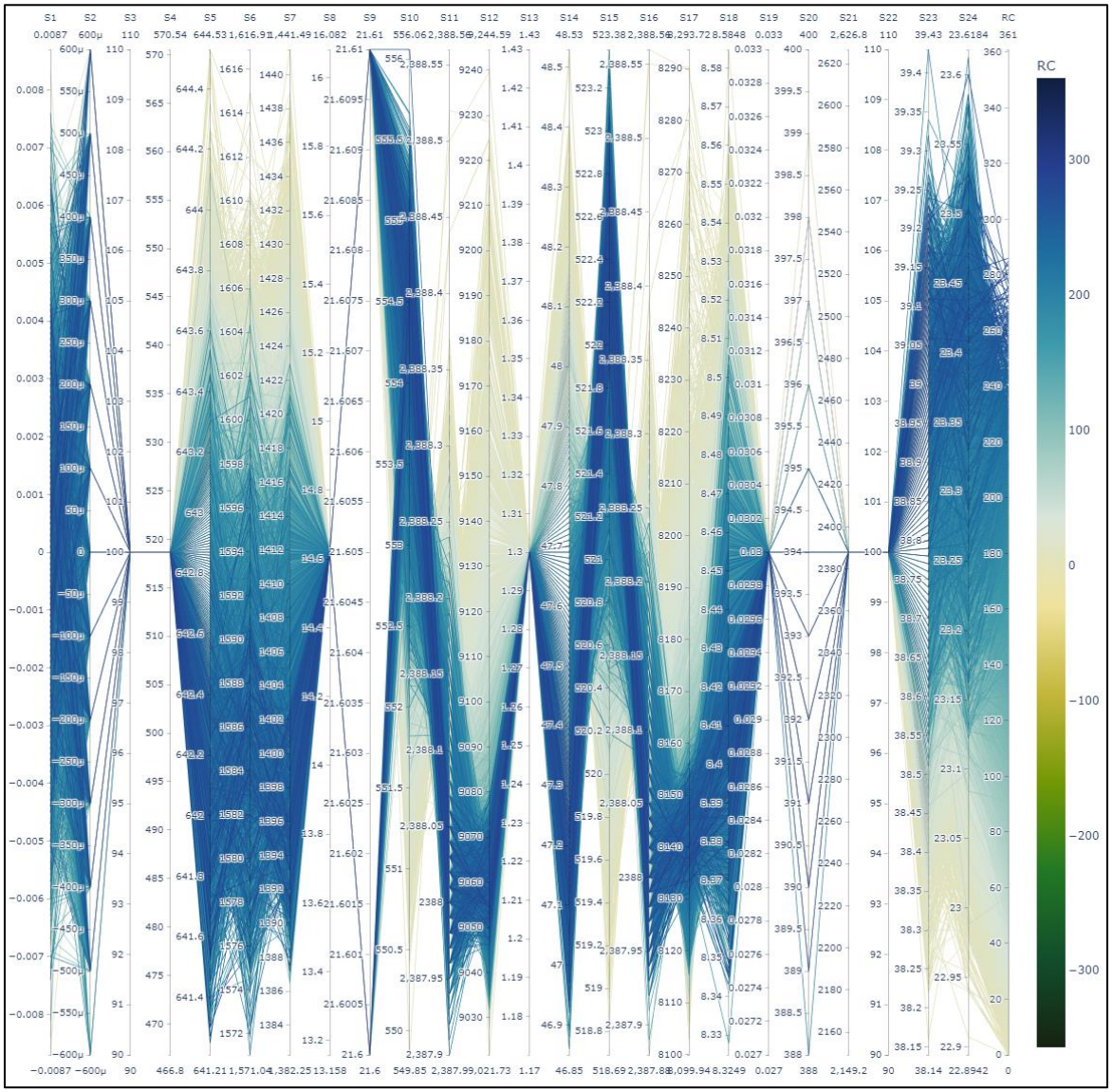


Fig. Parallel Coordinates plot for CMAPSS Dataset

In the initial cycles, indicated by dark blue lines, the sensor readings tend to be more consistent and clustered together, signifying stable operation. As the engines approach mid-life, sensor readings start to diverge, reflecting variations in performance and the emergence of faults. Nearing the end of life, shown by yellow lines, the sensor readings show significant divergence, indicating wear and degradation. Some sensors exhibit considerable variations, highlighting their importance in predicting the remaining useful life of the engines, while others show minimal variation, suggesting they might be less critical for this task.

Occasionally, lines deviate significantly from the general pattern, indicating potential anomalies or sudden faults.

Data Pre-processing:

Some features are constant in the dataset, and thus, their variance is zero. All zero variance variables are removed before the training stage because they do not contain useful information for machine learning.

A) Data Filtering: The sensor data in the dataset are noisy and sporadic, necessitating the application of smoothing filters to improve data quality. Two widely used smoothing techniques, Simple Moving Average (SMA) and Exponential Moving Average (EMA), were applied with various weights to determine the most effective method. Upon evaluation, EMA with an alpha value of 0.1 visually outperformed other configurations. Consequently, EMA with this alpha value was selected and applied to the sensor data, resulting in a smoother and more reliable data.

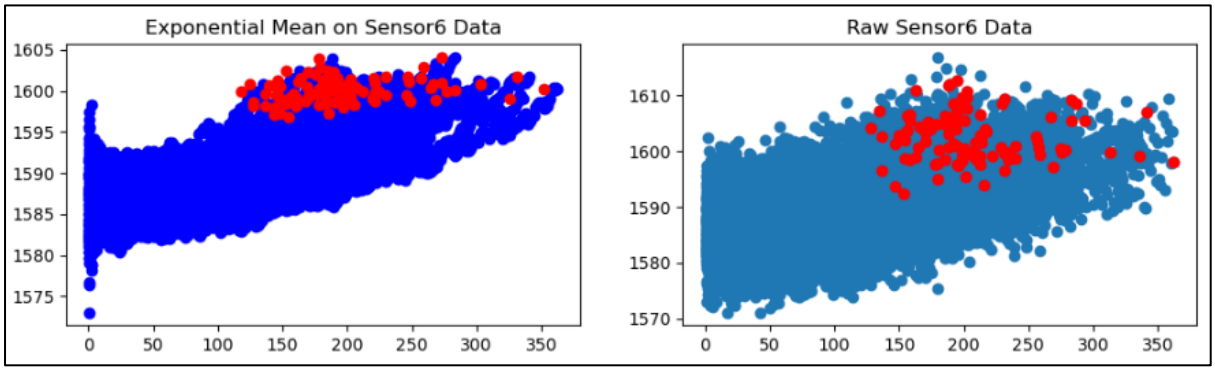


Fig. Comparison between Raw sensor data and exponential moving averaged sensor data

B) Data Scaling: Since the value range is substantially different in different variables, it can be difficult to find the optimal point for the cost function. Therefore, the training and testing datasets need to be normalized. There are two widely used methods for normalization, which are Z-scores (Equation (1)) and min-max-scale (Equation (2)). Both methods are applied, and the one with the best evaluation result is selected.

$$x' = \frac{x - \text{mean}(x)}{\text{std}(x)} \quad (1)$$

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (2)$$

Principal Component Analysis:

As part of feature engineering, Principal Component Analysis (PCA) is applied to visualize and understand the high-dimensional sensor data. PCA is a widely used dimensionality reduction technique that transforms the data by projecting it onto a set of orthogonal axes. This method works by identifying the eigenvectors and eigenvalues of the covariance matrix of the dataset. The eigenvectors, known as the "Principal Components," represent the directions of maximum variance in the data. By projecting the data onto these principal components, PCA reduces the dimensionality while retaining the most significant features, facilitating better visualization and analysis of the complex sensor data.

Explained Variance Ratio: The number of components needed can be determined by looking at the cumulative explained variance ratio as a function of the number of components as shown in the below graph.

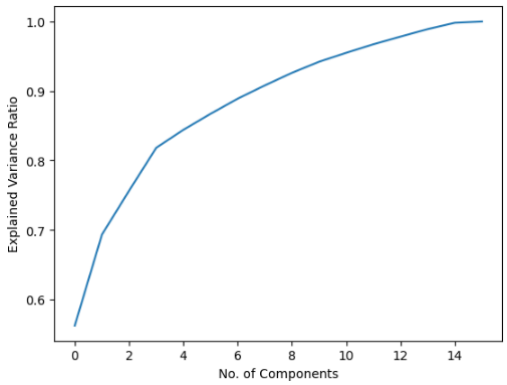


Fig. Explained Variance Ratio of Principal Components

This curve quantifies how much of the total, the high dimensional variance is contained within the first N components. For example, we see that the first two components contain approximately 70% of the variance, while we need 12 components to describe close to 100% of the variance. Which means we can reduce our data dimension to 24 from 12 without much loss of the data.

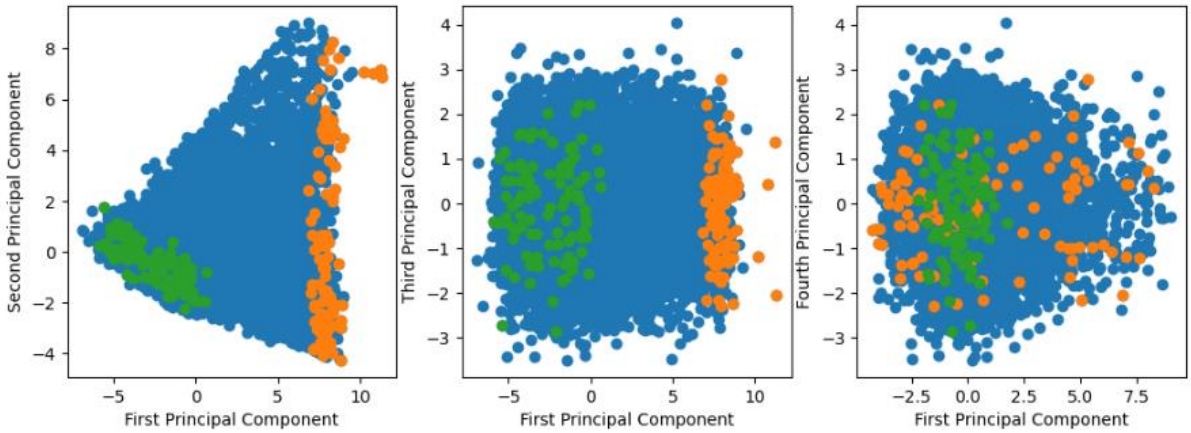


Fig. First 3 principal components are plotted

The first, second, and third principal components are studied extensively as they represent 75% of the dataset's variance. The above figure represents the scatter plot for the complete dataset, where green dots represent the starting points of the engine cycles, and orange dots represent the failure points. It becomes evident from the scatter plot that thresholding the first principal component can effectively determine the failure point of all engines, irrespective of the other principal components.

Feature Selection:

From the plots shown in the figure below, it is evident that the first principal component can be segregated to indicate failure, which helps the model in tuning the remaining useful life (RUL). Therefore, the first principal component is included as an additional feature for model development. This inclusion enhances the model's ability to

predict RUL more accurately by leveraging the significant variance captured by the first principal component.

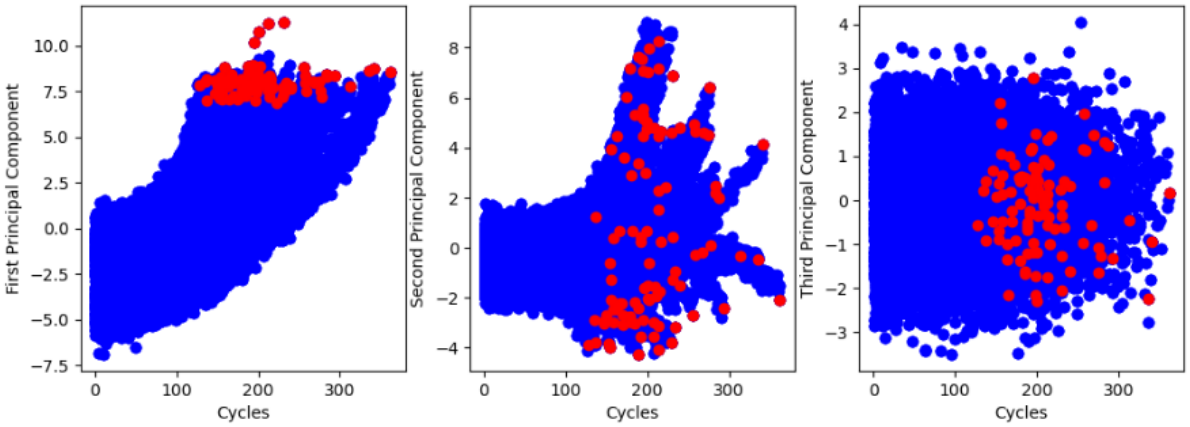


Fig Principal Components Vs Cycles

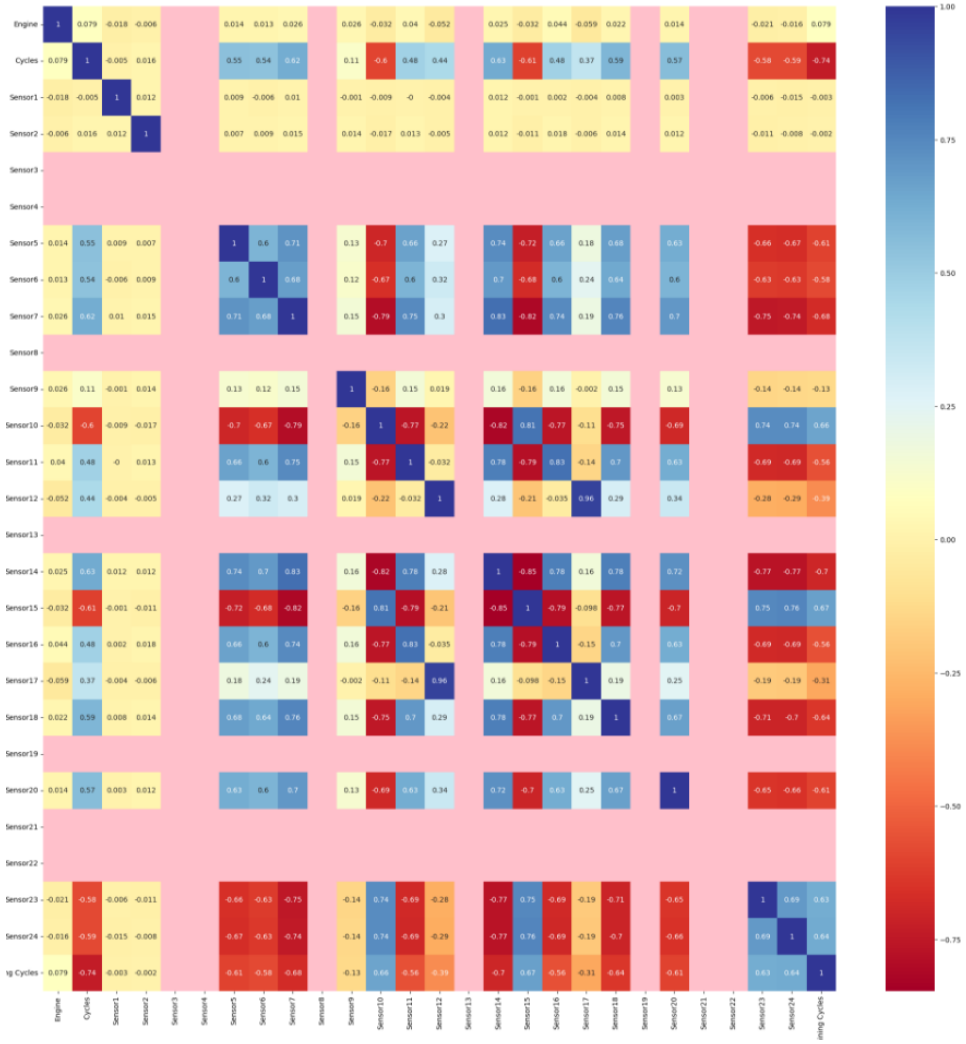


Fig. Heatmap of the features in CMAPSS dataset

From the heat map it is evident that lot of features are highly correlated and it can lead to multi collinearity problem. Multicollinearity refers to the situation where two or more features in a dataset are highly correlated, which can negatively impact the performance of machine learning models. In the presence of multicollinearity, the model

coefficients can become unstable, leading to high variance and unreliable predictions. Therefore, it is crucial to select features that contribute the most to the model's performance while minimizing redundancy. To tackle the issue of multicollinearity and ensure the selection of the most relevant features, we applied the Select K Best algorithm. This algorithm ranks all the features according to a specified statistical criterion and selects the top K features that are most significant for predicting the target variable. From the below graph it is evident that Sensor1 and Sensor2 are less significant and hence, it can be removed.

Features	Scores
PC1	135891.183009
Sensor14	19406.598613
Sensor7	17641.647140
Sensor15	16985.048173
Sensor10	15685.929943
Sensor18	14515.353596
Sensor24	13987.262919
Sensor23	13535.168228
Sensor5	12002.674101
Sensor20	11982.004911
Sensor6	10706.067107
Sensor11	9621.478047
Sensor16	9551.698672
Sensor12	3702.793868
Sensor17	2143.008595
Sensor1	0.211040
Sensor2	0.078251

Fig. Features ranked based on Scores by Select K Best Algorithm

Model Selection:

According to our literature survey, Remaining Useful Life (RUL) prediction performs particularly well using neural network-based deep learning algorithms. To identify a highly robust model in terms of performance, we compared the results of several machine learning and deep learning models in our work. These models include Linear Regression, Random Forest Regression, eXtreme Gradient Boosting (XGBoost), Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM) networks, and a hybrid model combining Convolutional Neural Networks (CNN) with LSTM. This comprehensive comparison allows us to evaluate each model's effectiveness in predicting RUL and select the best-performing approach for our predictive maintenance application

Linear Regression Model: Linear regression assumes a linear relationship between the independent variables (predictors) and the dependent variable (response). It serves as a fundamental building block in many statistical and machine learning techniques. Linear regression is applied on the data and the performance is evaluated based on RMSE and R2 scores for various hyperparameters. Additionally, to reduce overfitting, L1 and L2 regularisation are applied.

Random Forest Model: Random Forest Regressor is an ensemble learning method used for regression tasks. It is an extension of the Random Forest algorithm [6], which is originally designed for classification. In regression, the goal is to predict a continuous target variable. The Random Forest Regressor constructs a large number of decision trees during training, each tree trained on a random subset of the data. The final prediction for a new sample is obtained by averaging the predictions from all individual trees in the forest. This helps reduce overfitting by creating trees that are different from each other.

The following hyperparameters were explored using grid search, and the best estimator was selected based on the mean squared error (MSE)

Hyperparameter	*Description*	*List of Values*	***Best Estimator***
n_estimators	No. of Trees	[100,200,300]	**300**
max_depth	Maximum depth of trees	[6,8,10,12,14]	**6**
min_samples_leaf	Minimum of samples for leaf	[4,6,8,10]	**4**
ccp_alpha	Tree pruning factor	[0,1,2]	**0**

XGBoost Model: *XGBoost* Regressor is a powerful and efficient implementation of the gradient boosting algorithm specifically designed for regression tasks. XGBoost (extreme Gradient Boosting) [7] is an ensemble of decision trees sequentially. Each new tree corrects the errors made by the previous trees, effectively reducing bias and variance. XGBoost is designed for efficiency, utilizing advanced optimization techniques and parallel processing to handle large datasets and complex tasks quickly. It also incorporates [8] both L1 (Lasso) and L2 (Ridge) regularization in its objective function to prevent overfitting and improve generalization. While also using maximum depth features, ensuring trees grow only as deep as necessary to prevent overfitting and improve computational efficiency.

The following hyperparameters were explored using grid search, and the best estimator was selected based on the mean squared error (MSE)

Hyperparameter	*Description*	*List of Values*	***Best Estimator***
learning_rate	Learning Rate	[0.05,0.1,0.2]	**0.1**
n_estimators	No. of Trees	[70,100,200,300]	**70**
max_depth	Maximum depth of trees	[3,4,5]	**3**
min_child_weight	Minimum sum of instance weights	[70,100,150,200]	**200**

Multi-Layer Perceptron Model: A Multilayer Perceptron Regressor (MLPRegressor) is a type of artificial neural network used for regression tasks. Unlike traditional linear regression models, MLPRegressor can capture complex, non-linear relationships in the data, making it a powerful tool for many regression problems. An MLP consists of an input layer [10], one or more hidden layers, and an output layer. Each layer contains a certain number of neurons, which are the basic units in each layer that apply a weighted sum of inputs followed by an activation function.

MLPRegressor trains iteratively since at each time step the partial derivatives of the loss function with respect to the model parameters are computed to update the parameters

[9]. The MLP Regressor showed no overfitting in the initial stage of experimentation. But when the training and testing values were clipped at 130, it overfits. However, the testing accuracy was still higher than all the previous models.

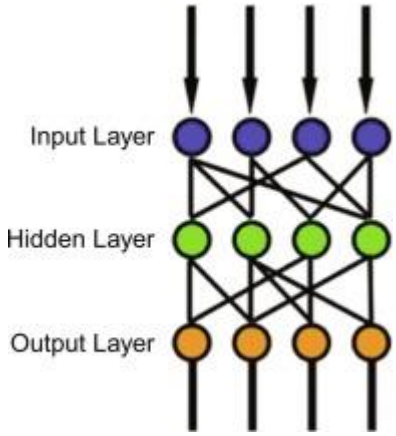


Fig. Multilayer Perceptron Flow Diagram

The MLP model underwent fine-tuning, where various combinations of hidden layer sizes were tested across different solvers. After evaluation, the optimal estimator was determined based on specific parameter configurations.

LSTM Model: Long Short-Term Memory Network (LSTM) [5] is a type of RNN network for sequence learning tasks and has achieved great success on speech recognition and machine translation. LSTM does not have long-term time dependency problems by controlling information flow using input gate, forget gate and output gate. Remembering information for long periods of time is practically their default behaviour. Due to inherent sequential nature of sensor data, LSTMs are well-suited for RUL estimation using sensor data. In this work, we propose a LSTM based approach for RUL estimation, which uses multiple layers of LSTM cells in combination with standard feed forward layers to discover hidden patterns from sensor and operational data with multiple operating conditions, fault and degradation models.

LSTM cell structure at time t is shown in Figure 2 [5]. We differentiate output of LSTM cell and cell state denoted as h (t) and c (t), respectively. Vector size of the output and cell state is the same and it is defined by number of nodes in the cell. This LSTM cell also takes sensor data x (t) as an input. There are three gates that control the information flow within cell:
(1) Input gate it controls what information based on output h (t-1) and sensor measurements x (t) will be passed to memory cell,
(2) Output gate controls what information will be carried to the next time step, and
(3) Forget gate controls how memory cell will be updated as shown in the below Figure.

There are many variants of LSTMs and experimentation on this is part of our future work. For example, the nonlinear sigmoid and hyperbolic tangent activation function can be replaced using other activation functions. One can also choose to use the cell state c (t-1) as an extra input into the three gates. Connection between different layers of LSTMs in Figure 2 is achieved such that the output of one layer is as an input to the next layer. Sensor measurements are input only to the first LSTM layer.

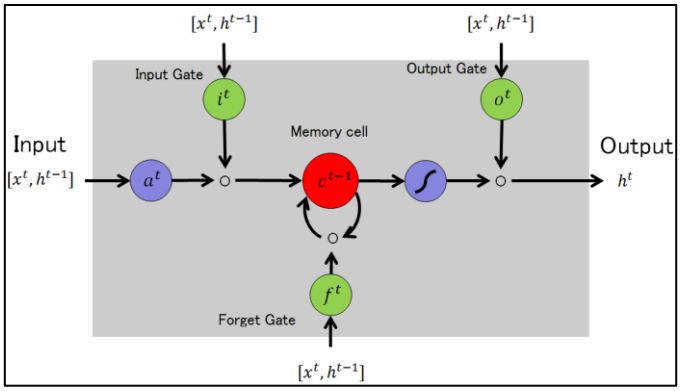


Fig. LSTM Cell

LSTM works only on time series data, hence the dataset is split into multiple fixed length time series data of length 30 and applied to the model.

CNN + LSTM: The combination of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks has gained significant attention in the field of time series forecasting and sequence modeling. This hybrid architecture leverages the strengths of both CNNs and LSTMs to capture spatial patterns and long-term dependencies in sequential data. The CNN module processes the input data (e.g., time series sequences) and extracts relevant spatial features through convolutional layers. These layers employ filters to capture local patterns within the input sequences, enabling the model to learn hierarchical representations of the data. The LSTM module receives the output from the CNN module and processes it sequentially to capture long-term dependencies and temporal patterns. The LSTM layers maintain an internal state that allows them to remember information over extended time periods, making them well-suited for modeling sequential data.

Model Evaluation:

In order to evaluate the performance of a RUL estimation model on the test data, Root Mean Square Error (RMSE) (Equation 2), gives equal penalty weights to the model when the estimated RUL is smaller than true RUL and when the estimated RUL is larger than true RUL and R2 Score (Equation 1), which is also widely used as an evaluation metric for the estimation of RUL.

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} \qquad (1)$$

$$\text{RMSD} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - x_0)^2}. \qquad (2)$$

All the models are evaluated based on RMSE and R2 score on the test dataset and the results are shown in the table below.

Model	*Test RMSE*	*Test R2 Score*
Linear Regression	43.18	0.46
Random Forest	6.68	0.42
XG Boost	17.35	0.65
Multilayer Perceptron	**4.51**	0.52
LSTM	15.93	0.75
CNN + LSTM	13.34	**0.86**

Flask Application:

A web application is created using Flask module with the following 2 pages.

Pages:

Main Page: As shown in Fig1, the page contains a button labeled "Predict RUL". Upon clicking this button, the application loads an inbuilt test set, performs data transformation, scaling, and executes multiple predictive maintenance models.

Results Page: As shown in Fig2, After the successful execution of all models, the application redirects to the results page. This page displays bar plots for each model's predictions.

Functionality:

Predict RUL Button: When clicked, this button triggers the execution of the predictive maintenance models.

Data Transformation and Scaling: Before model execution, the application preprocesses the data by transforming and scaling it appropriately for each model.

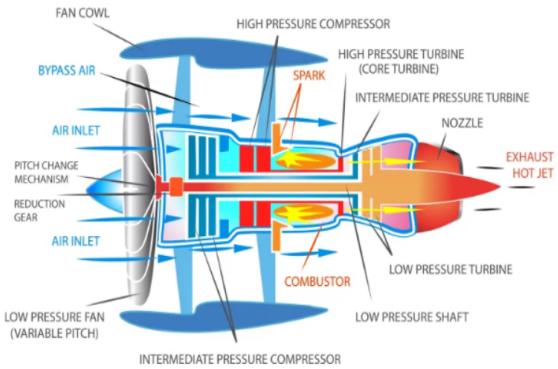
Model Execution: The application executes multiple predictive maintenance models, including LSTM, XGBoost, MLP, and CNN with LSTM.

Bar Plot Visualization: Once all models have been executed successfully, the application generates bar plots for each model's predictions on the results page.

Workflow:

- User navigates to the main page of the web application.
- User clicks the "Predict RUL" button.
- The application loads the inbuilt test set and preprocesses the data.
- Multiple predictive maintenance models (LSTM, XGBoost, MLP, CNN with LSTM) are executed.
- After successful execution, the application redirects to the results page.
- On the results page, bar plots for each model's predictions are displayed.

Predictive Maintenace of NASA Aircraft Engine



Predict RUL

The degradation that can occur to aircraft parts over the course of their lifetime directly impacts both their performance and reliability. In order to provide the necessary maintenance behavior, this machine learning research will be focused on providing a framework for predicting the aircraft's remaining usable life (RUL) based on the whole life cycle data. Using the NASA C-MAPSS data set is implemented and tested to determine the engine lifetime. Tracking and predicting the progression of damage in aircraft engine turbo machinery has some roots in the work of Kurosaki. They estimate the efficiency and the flow rate deviation of the compressor and the turbine based on operational data, and utilize this information for fault detection purposes. A low pressure compressor (LPC) and high pressure compressor (HPC) supply compressed high temperature, high pressure gases to the combustor. Low pressure turbine (LPT) can decelerate and pressurize air to improve the chemical energy conversion efficiency of aviation kerosene. High pressure turbines (HPT) generate mechanical energy by using high temperature and high pressure gas strike turbine blades. Low-pressure rotor (N1), high-pressure rotor (N2), and nozzle guarantee the combustion efficiency of the engine. Our main focus will be on accurately recording low RUL values to prevent putting the engine at danger and forecasting the RUL of the turbofan engine while accounting for HPC failure. Data analysis, data visualization and Model development are covered in this project

Fig1. Main Page

Predictive Maintenace of NASA Aircraft Engine

Remaining Useful Life Prediction

The NASA Aircraft Engine dataset has been extensively analyzed to predict the remaining useful life (RUL) using various machine learning and deep learning models. Among the models tested, the Multilayer Perceptron (MLP), XGBoost, Long Short-Term Memory (LSTM), and a hybrid version of LSTM combined with Convolutional Neural Networks (CNN) have shown particularly convincing results. These models have demonstrated superior performance in terms of accuracy and robustness in predicting RUL, making them standout choices for this application

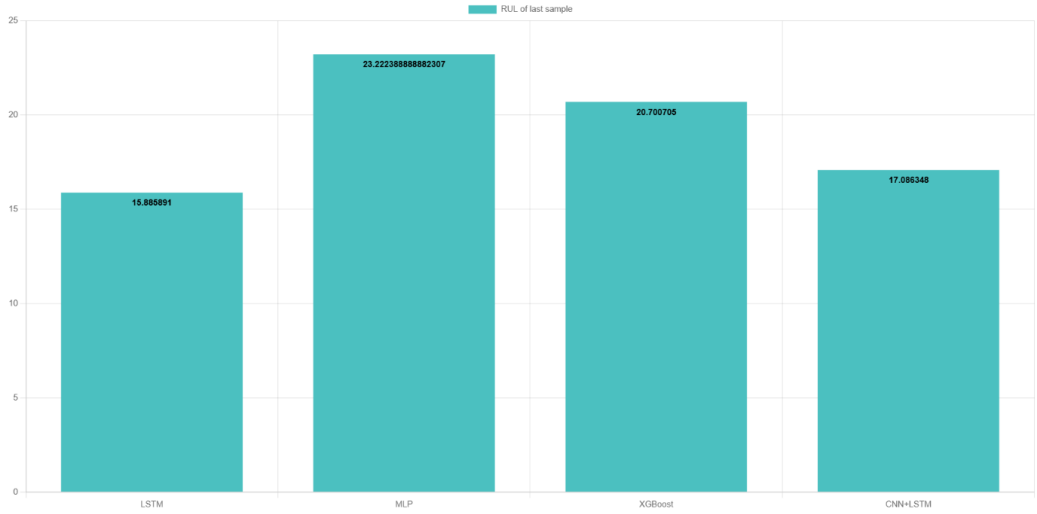


Fig2. Results Page

Conclusion:

The project encompassed the entire lifecycle of predictive maintenance, starting with data collection from various sources, followed by comprehensive data preparation and preprocessing stages to ensure data quality and consistency. Feature scaling, Principal Component Analysis (PCA), and feature selection techniques were applied to refine the dataset and identify the most relevant features for modeling. Multiple predictive maintenance models, including LSTM, XGBoost, MLP, and CNN with LSTM, were developed and evaluated using appropriate metrics. Finally, a user-friendly web application was developed using Flask, providing an intuitive interface for users to interact with the models. From data input to model execution and result visualization, the application seamlessly facilitated predictive maintenance tasks, empowering maintenance professionals to optimize equipment uptime and reduce downtime costs effectively.

References:

[1] S. Duffuaa, M. Ben-Daya, K. Al-Sultan, and A. Andijani, “A generic conceptual simulation model for maintenance systems,” Journal of Quality in Maintenance Engineering, vol. 7, pp. 207–219, 09 2001.

[2] X.-S. Si, W. Wang, C.-H. Hu, and D.-H. Zhou, “Remaining useful life estimation–a review on the statistical data driven approaches,” European Journal of Operational Research, vol. 213, no. 1, pp. 1–14, 2011

[3] Heimes, F.O.: Recurrent neural networks for remaining useful life estimation. In: International Conference on Prognostics and Health Management, 2008. PHM 2008.

[4] G. S. Babu, P. Zhao, and X.-L. Li, “Deep convolutional neural network based regression approach for estimation of remaining useful life,” in International Conference on Database Systems for Advanced Applications. Springer, 2016, pp. 214–228

[5] Shuai Zheng, Kosta Ristovski, Ahmed Farahat and Chetan Gupta, “Long Short-Term Memory Network for Remaining Useful Life Estimation,” in IEEE International Conference on Prognostics and Health Management (ICPHM), 2017

[6] Breiman, L. (2001). "Random Forests." Machine Learning, 45(1), 5-32.

[7] Chen, T., & Guestrin, C. (2016). "XGBoost: A Scalable Tree Boosting System." In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[8] XGBoost Documentation

[9] Scikit-learn Documentation: MLPRegressor.

[10] S. Abirami and P. Chitra, "The Digital Twin Paradigm for Smarter Systems and Environments: The Industry Use Cases," in Advances in Computers,2020.

[11] Wegman, E. J. (1990). "Hyperdimensional Data Analysis Using Parallel Coordinates," Journal of the American Statistical Association, vol. 85.